INDEX

Name of the Laboratory :					
Sub. Code :					
Name of the Staff In-Charge :					
Sl No	Date of Experiment	Name of the Experiment	Page No	Date of Submission	Staff Signature with Date
1	27/06/2018	Creation of Date And Time server	1		
2	04/07/2018	Print the client's Address on the server	5		
3	11/07/2018	Creation of UDP Server	8		
4	18/07/2018	Creation of Chat program	12		
5	25/07/2018	Calculation of Checksum for Packet Date & File	15		
6	29/08/2018	Program to implement HTTP Protocol	20		
7	05/09/2018	Creation of Telnet Protocol	22		
8	05/09/2018	Implement FTP using TCP	25		
9	05/09/2018	Implement FTP using UDP	28		
10	05/09/2018	Router Configuration	31		

Expt. No. 01	Page No. 01
Expt. Name. Creation of Date And Time Sever	Date. 27/06/18

Aim: Program to print date and time in client by server.

Alogorithm:

SERVER

STEP1: Create instances for socket and ServerSocket class.

STEP2: Use the port 8020 for TCP.

STEP3: Make the PrintStream object connect to the OuputStream using Socket.

STEP4: Create an instance of the Date class and write it into the Socket.

STEP5: Get the IP address of the client using the socket and getInetAddress().

CLIENT

STEP1: Create instances for socket class with the port number 8020.

STEP2: Create an object of DataInputStream and make it to get data from server through the socket.

STEP3: Read the Date object.

STEP4: Print the obtained date.

Program:

server:

```
import java.net.*;
import java.io.*;
import java.util.*;
public class server2 extends Thread
{
public static void main(String[] args) throws Exception
{
ServerSocket sSocket = new ServerSocket(1000);
Socket cSocket=sSocket.accept();
BufferedReader br=new BufferedReader(new
InputStreamReader(cSocket.getInputStream()));
```

```
PrintWriter out=new PrintWriter(cSocket.getOutputStream(),true);
Date d = new Date();
try
    {
    while(true)
    {
    d= new Date();
out.println("Time at server;" +d.toString());
System.out.println(br.readLine());
sleep(1000);
}
}
catch(IOException e)
    {
System.out.println("----Client has Closed-----");
    }
}
}
```

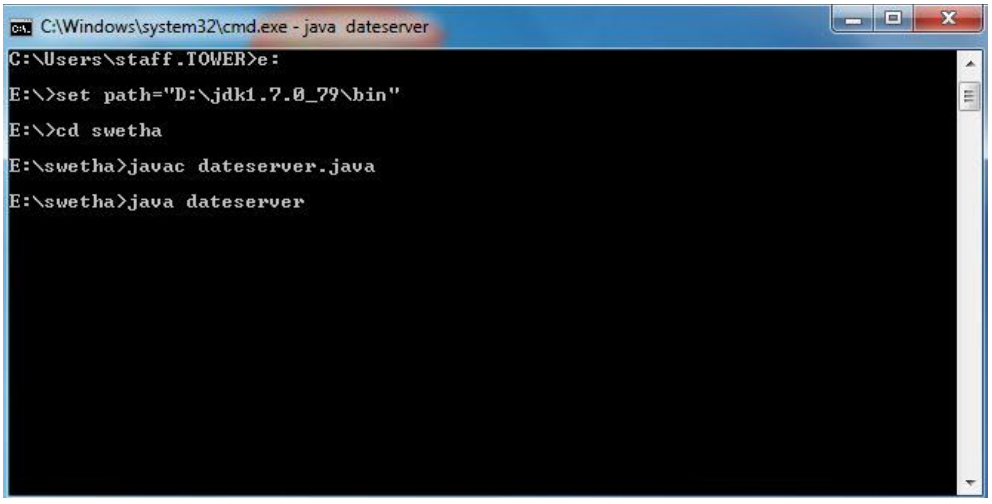
Client:

```
import java.net.*;
import java.io.*;
public class client2
{
public static void main (String[] arg) throws Exception
{
try
{

Socket s=new Socket(InetAddress.getLocalHost(),1000);
BufferedReader br = new BufferedReader(new
InputStreamReader(s.getInputStream()));
String input;
PrintWriter out=new PrintWriter(s.getOutputStream(),true);
```

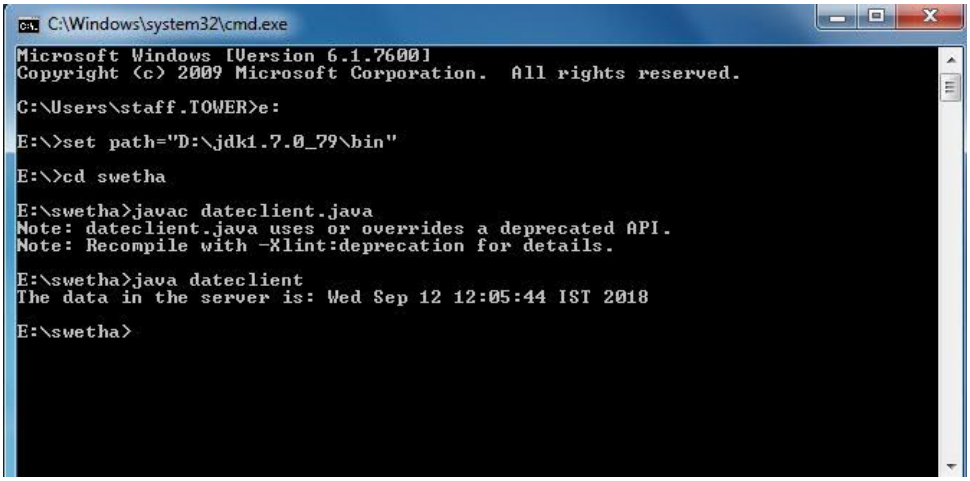
```
while((input=br.readLine())!=null)
{
System.out.println(input);
out.println("Date and Time Received----------client Acknowledge------");
}
}
catch(Exception e)
      {
      }
}
}
```

Output:

Server:



Client:



Result: The above program is executed and output was verified.

Expt. No. 02	Page No. 05
Expt. Name. Print the client's Address on the sereve.	Date. 04/07/18

Aim: Program to print client address at server side.

Algorithm:

SERVER

STEP1: Create instances for socket and ServerSocket class.

STEP2: Use the port 9000 for TCP.

STEP3: Make the PrintStream object connect to the OuputStream using Socket.

STEP4: Create an instance of the Date class and write it into the Socket.

STEP5: Get the IP address of the client using the socket and getInetAddress ().

STEP6: Print the client's IPAddress.

CLIENT

STEP1: Create instances for socket class with the port number 9000.

STEP2: Create an object of DataInputStream and make it to get data from server through the socket.

STEP3: Read the Date object.

STEP4: Print the obtained date.

Program:

Server:

```
import java.net.*;
import java.io.*;
class server4
{
public static void main(String[] args) throws Exception
{
ServerSocket s1 = new ServerSocket(8000);
System.out.println("Server Running");
Socket s2=s1.accept();
InetAddress a=InetAddress.getLocalHost();
```

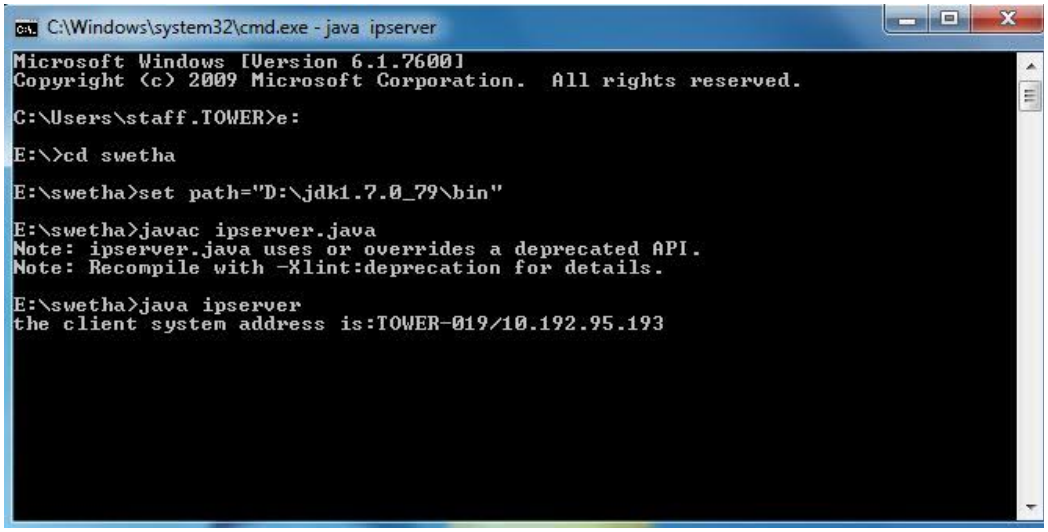
```
        String add=a.getHostAddress();
        BufferedReader in=new BufferedReader(new
        InputStreamReader(s2.getInputStream()));
        PrintWriter out=new PrintWriter(s2.getOutputStream(),true);
        System.out.println("Client Connected");
        System.out.println(s2);
        out.println(add);
        System.out.println("Client's IP is ");
        System.out.println(in.readLine());
        s1.close();
        s2.close();
        }
        }
```

Client:

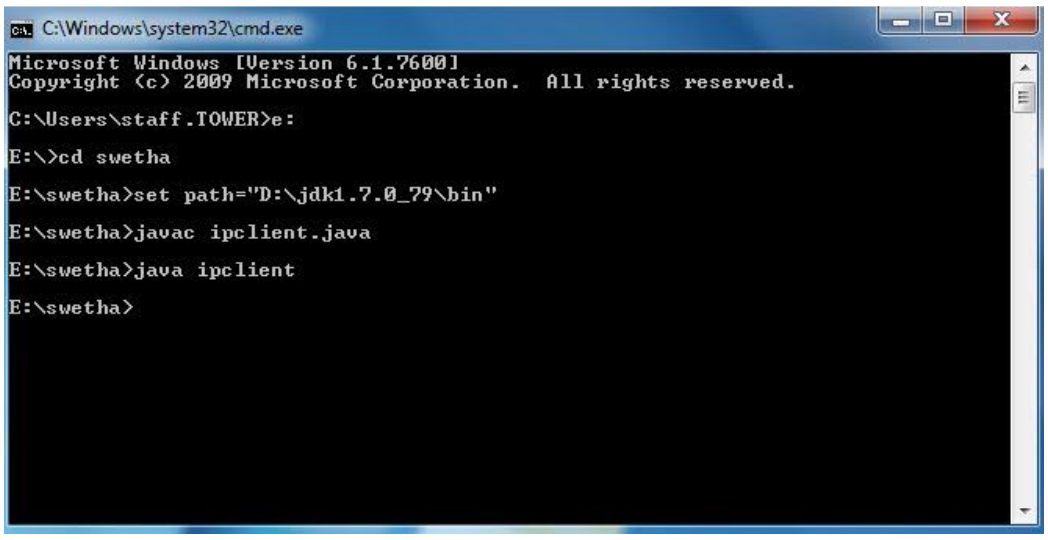
```
        import java.net.*;
        import java.io.*;
        class client4
        {
        public static void main(String[] args) throws Exception
        {
        InetAddress a=InetAddress.getLocalHost();
        Socket s2 = new Socket(a,8000);
        String add=a.getHostAddress();
        BufferedReader in=new BufferedReader(new
        InputStreamReader(s2.getInputStream()));
        PrintWriter out=new PrintWriter(new
        OutputStreamWriter(s2.getOutputStream()),true);
        System.out.println(in.readLine());
        out.println(add);
        s2.close();
        }
        }
```

Output:

Server:



Client:



Result: The above program is executed and output was verified

Expt. No. 03	Page No. 08
Expt. Name. Creation of UDP server	Date. 11/07/18

Aim: To perform a java program for UDP client and server.

Algorithm:

SERVER:

1.Create a new Datagram Socket.

2.Create a new Datagram packet.

3.Create a message to be sent.

4.Convert into bytes

5.create a packet

6.send packet

7.wait for acknowledgement from client

8.print data from client

9.stop the program

CLIENT:

1. Create new Datagram Socket.

2.Create new Datagram packet.

3.Get the packet.

4.Print the content.

5.Create a new packet.

6.send to server

7.Stop the program.

Program:

Server:

```
import java.net.*;
import java.io.*;
public class udpserver
{
public static int client=789;
public static int server=790;
public static void main(String arg[]) throws IOException
{
String s;
InetAddress id=InetAddress.getLocalHost();
BufferedReader dis=new BufferedReader(new InputStreamReader(System.in));
DatagramSocket ds=new DatagramSocket(server);
byte b[]=new byte[1024];
System.out.println("Server Side.... Sending....");
System.out.println("\n"+id);
while(true)
{
s=dis.readLine();
if(s.equals("end"))
{
b=s.getBytes();
DatagramPacket dp=new DatagramPacket(b,s.length(),id,client);
ds.send(dp);
break;
}
else
{
b=s.getBytes();
DatagramPacket dp=new DatagramPacket(b,s.length(),id,client);
ds.send(dp);
}
}
```

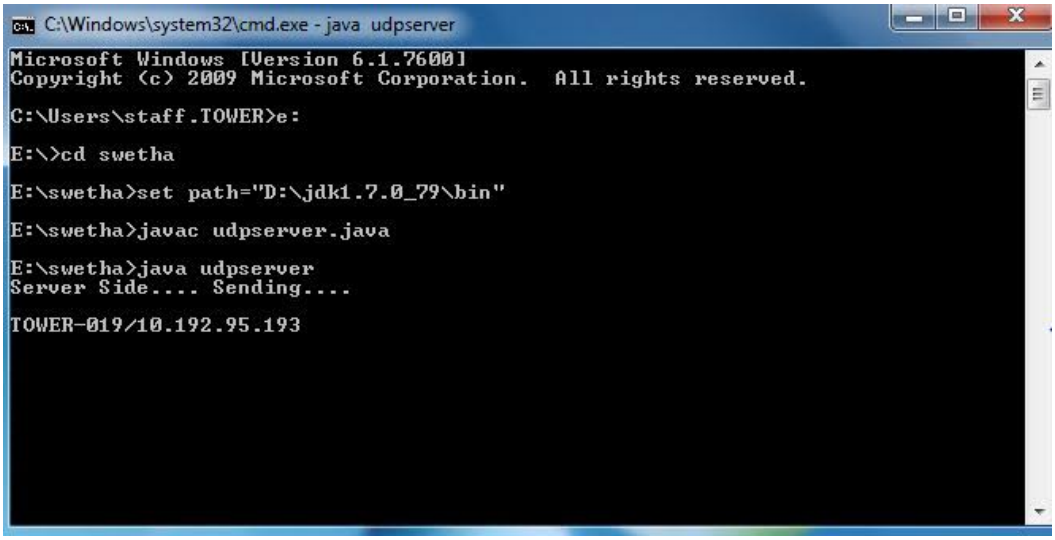
```
}
}
```

Client:

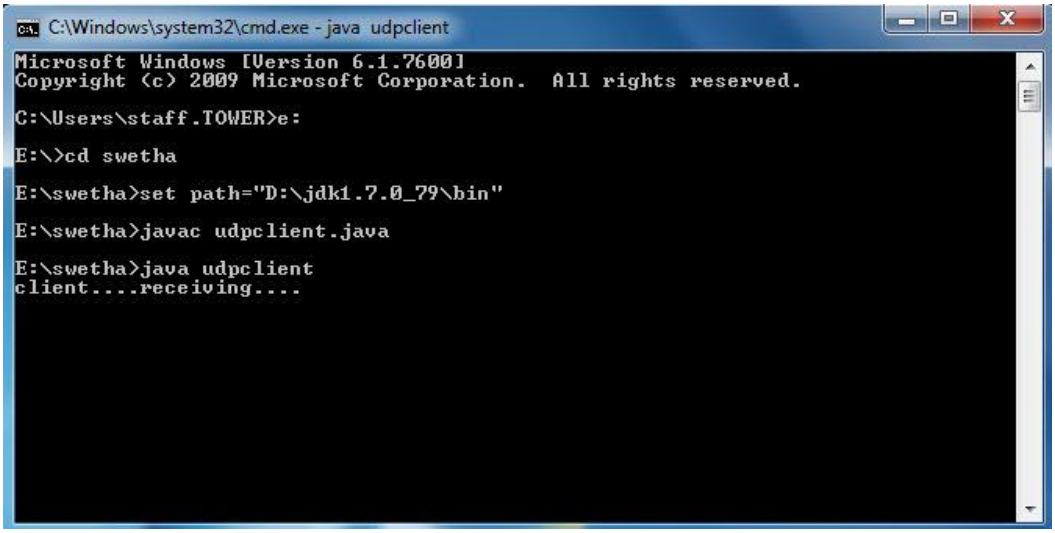
```
import java.net.*;
import java.io.*;
public class udpclient
{
public static int client=789;
public static void main(String args[]) throws IOException
{
DatagramSocket ds=new DatagramSocket(client);
byte b[]=new byte[1024];
System.out.println("client....receiving....");
while(true)
{
DatagramPacket dp=new DatagramPacket(b,b.length);
ds.receive(dp);
String s=new String(dp.getData(),0,dp.getLength());
if(s.equals("end")) break;
else System.out.println(s);
}
}
}
```

Output:

server:



Client:



Result: The above program is executed and output was verified.

Expt. No. 04	Page No. 12
Expt. Name. Creation of chat program	Date. 18/07/18

Aim: Program to create simple chat application.

Algorithm:

SERVER

STEP1: Instances of vector class is used to keep track of number of clients that can be connected and currently logged.
STEP2: The method that is responsible for sending the message to the clients is made synchronized.
STEP3: Server is capable of keeping into account the number of users. It adds and removes the client from the vector list as and when the connections are established and terminated.

CLIENT

STEP1: The client receives the name of the user and message of that user and sends it to client. Server then passes it on to all clients connected.

Program:

Server:

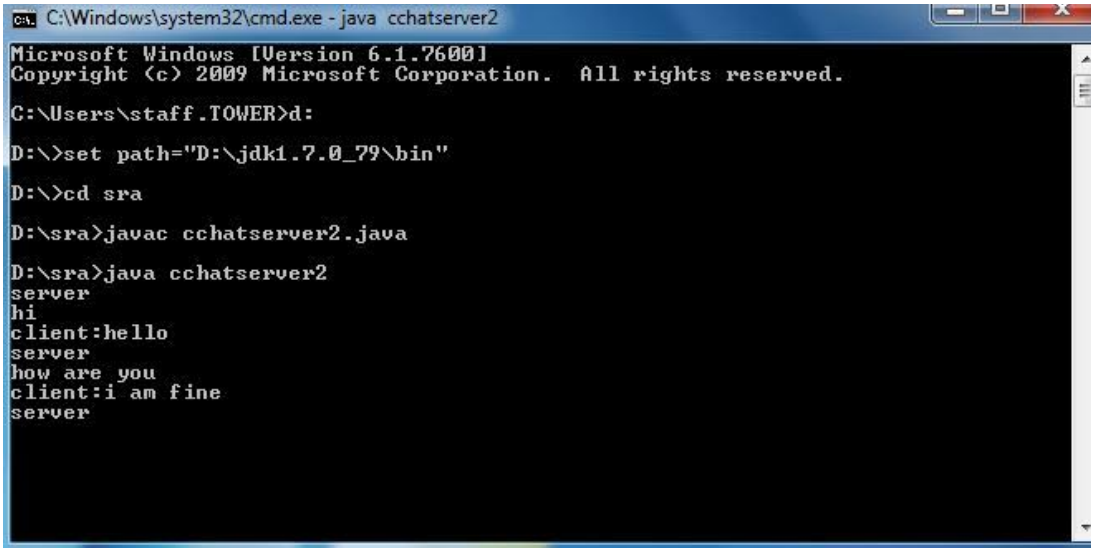
```
import java.net.*;
import java.io.*;
import java.util.*;
public class cchatserver extends Thread
{
public static void main(String arg[])throws Exception
{
ServerSocket ssocket=new ServerSocket(4000);
Socket csocket=ssocket.accept();
BufferedReader br=new BufferedReader(new
InputStreamReader(csocket.getInputStream()));
BufferedReader in=new BufferedReader(new InputStreamReader(System.in));
PrintWriter out=new PrintWriter(csocket.getOutputStream(),true);
String s,t;
try
{
```

```
        while(true)
        {
        System.out.println("server");
        s=in.readLine();
        out.println("server:"+s);
        System.out.println(br.readLine());
        }
        }
        catch(IOException e)
        {
        System.out.println("client has closed");
        }
        }
        }

client:
        import java.net.*;
        import java.io.*;
        public class cchatclient
        {
        public static void main(String arg[])throws Exception
        {
        Socket s=new Socket(InetAddress.getLocalHost(),4000);
        BufferedReader br=new BufferedReader(new
        InputStreamReader(s.getInputStream()));
        BufferedReader in=new BufferedReader(new InputStreamReader(System.in));
        PrintWriter out=new PrintWriter(s.getOutputStream(),true);
        String input,t;
        while(true)
        {
        System.out.println("client");
        out.println("client:"+in.readLine());
        System.out.println(br.readLine());
        }
        }
```

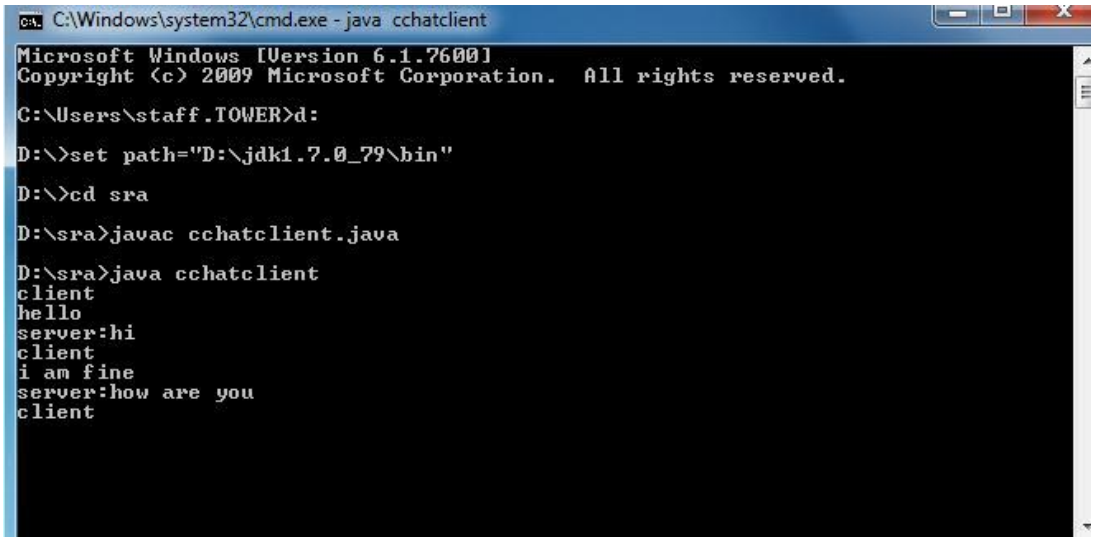
```
}
```

Output:

server:



Client:



Result: The above program is executed and output was verified.

Expt. No. 05	Page No. 15
Expt. Name. Calculation of checksum for packet date and file.	Date. 25/07/18

Aim: Program to perform checksum for Packet Date and file

Algorithm:

Server Side

1. Create objects of ServerSocket and Socket class
2. Create input and output buffers
3. Listen for client to connect
4. Get the data from client
5. Calculate the CRC
6. Get the CRC from client
7. If calculated CRC is same as that of received CRC, then send success message to client
8. Else send failure message
9. Close sockets
10. Stop

Client Side

1. Create object of Socket class and connect to the server
2. Send the data along with the CRC
3. Get the response from server and print
4. Close socket
5. Stop

Program:

Server:

```
import java.net.*;
import java.io.*;
import java.util.*;
import java.util.zip.*;
import java.lang.*;

public class Server extends Thread
{
public static void main(String[] args) throws Exception
   {
                try
                {
```

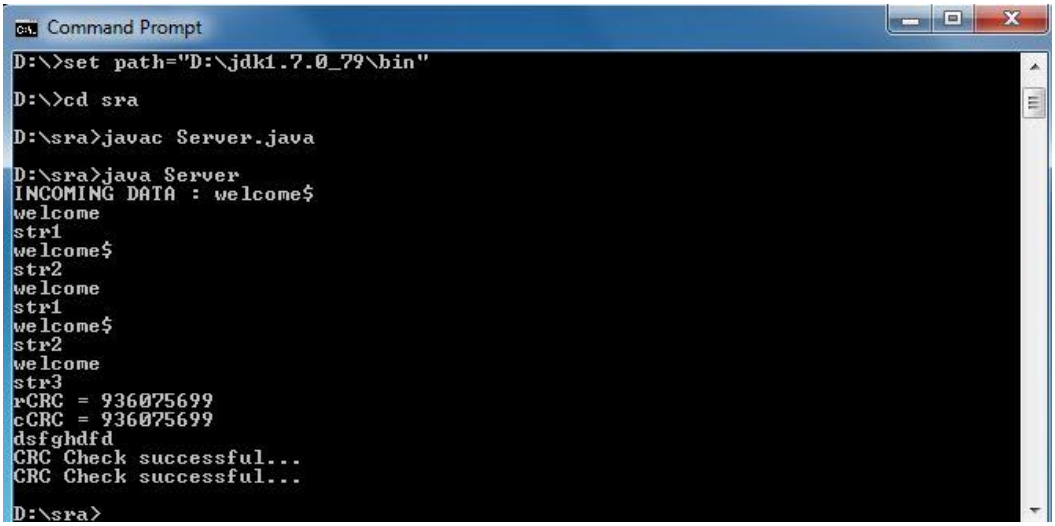
```
                ServerSocket sSocket=new ServerSocket(1000);
                Socket cSocket=sSocket.accept();
                BufferedReader br=new BufferedReader(new
        InputStreamReader(cSocket.getInputStream()));
                PrintWriter out=new
    PrintWriter(cSocket.getOutputStream(),true);
                String str1,str2;
                str2=br.readLine();
                System.out.println("INCOMING DATA : " + str2);
                StringTokenizer s=new StringTokenizer(str2,"$");
                CRC32 c=new CRC32();
                str1=s.nextToken();
                for(int i=0;i<s.countTokens()+2;i++)
                {
                        System.out.println(str1);

                        System.out.println("str1");

                        System.out.println(str2);
                        System.out.println("str2");
                }
                System.out.println(str1);
                c.update(str1.getBytes());
    System.out.println("str3");

                long rCRC=Long.parseLong(br.readLine()),cCRC=c.getValue();
                System.out.println("rCRC = " + rCRC);
                System.out.println("cCRC = " + cCRC);
    System.out.println("dsfghdfd");
                if(rCRC==cCRC)
                {
                        System.out.println("CRC Check successful...");
                        System.out.println("CRC Check successful...");
                }
                else
```

```
                        {
                                out.println("CRC Check un-successful...");
                                System.out.println("CRC Check un-successful...");
                        }
                }
                catch(Exception e){}
        }
    }

Client:
        import java.net.*;
        import java.io.*;
        import java.lang.*;
        import java.util.zip.*;

        public class Client
        {
        public static void main(String[] args) throws Exception
           {
                try
                {
                        Socket s=new Socket(InetAddress.getLocalHost(),1000);
                        BufferedReader br=new BufferedReader(new
        InputStreamReader(s.getInputStream()));
                        PrintWriter out=new PrintWriter(s.getOutputStream(),true);
                        String data="";
                        CRC32 c=new CRC32();

                        for(int i=0;i<args.length;i++)
                        {
                                System.out.println(args[i]);
                                c.update(args[i].getBytes());
                                data=data+args[i]+"$";
                        }
```

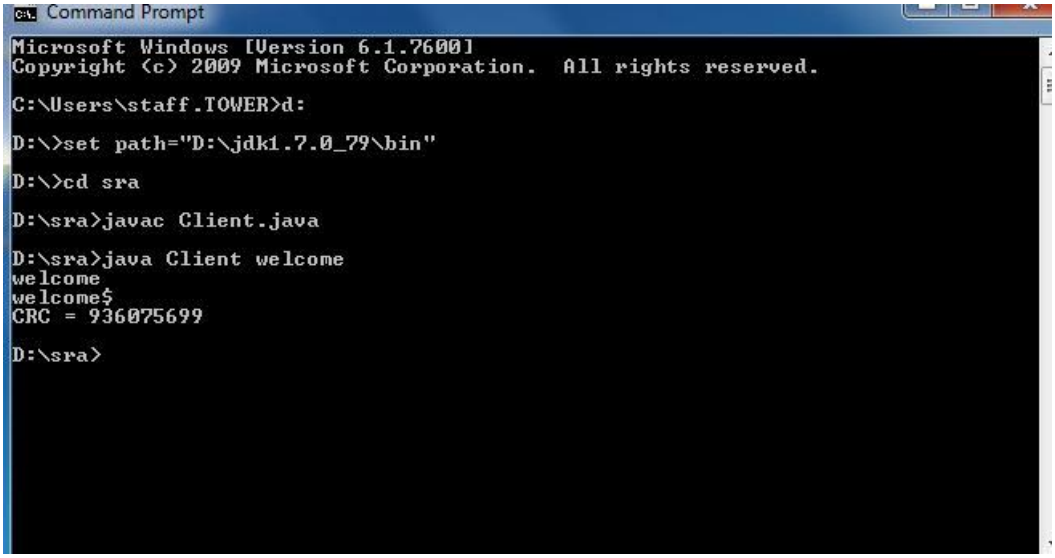
```
                    System.out.println(data);
                    System.out.println("CRC = " + c.getValue());
                    out.println(data);
                    out.println(c.getValue());
                    System.out.println(br.readLine());
                }
                catch(Exception e){}
    }
}
```

Output:

server:



Client:



Result: The above program is executed and output was verified.

Expt. No. 06	Page No. 20
Expt. Name. Program to implement HTTP protocol.	Date. 29/08/18

Aim: Program to implement HTTP protocol and to print URl for the Client.

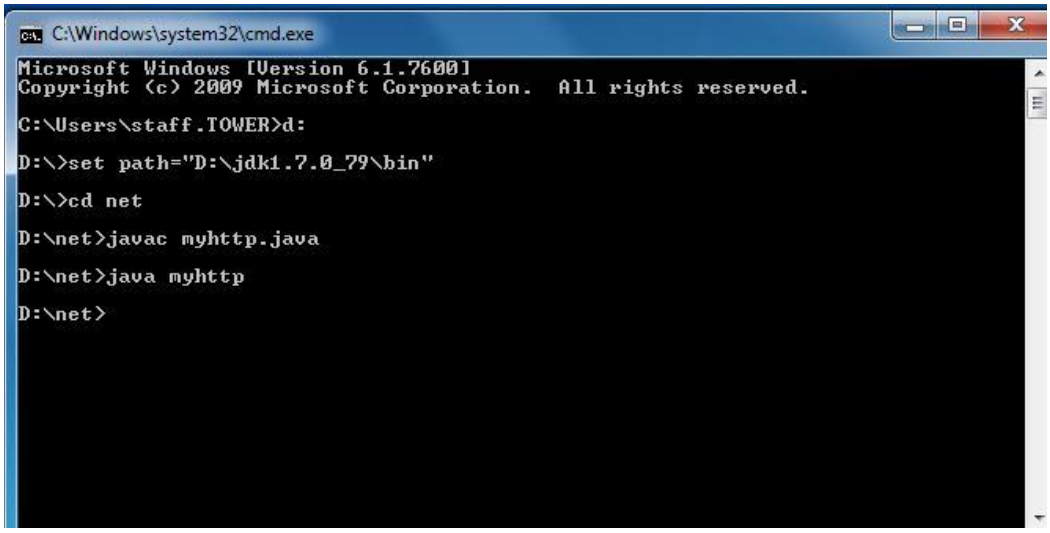
Algorithm:

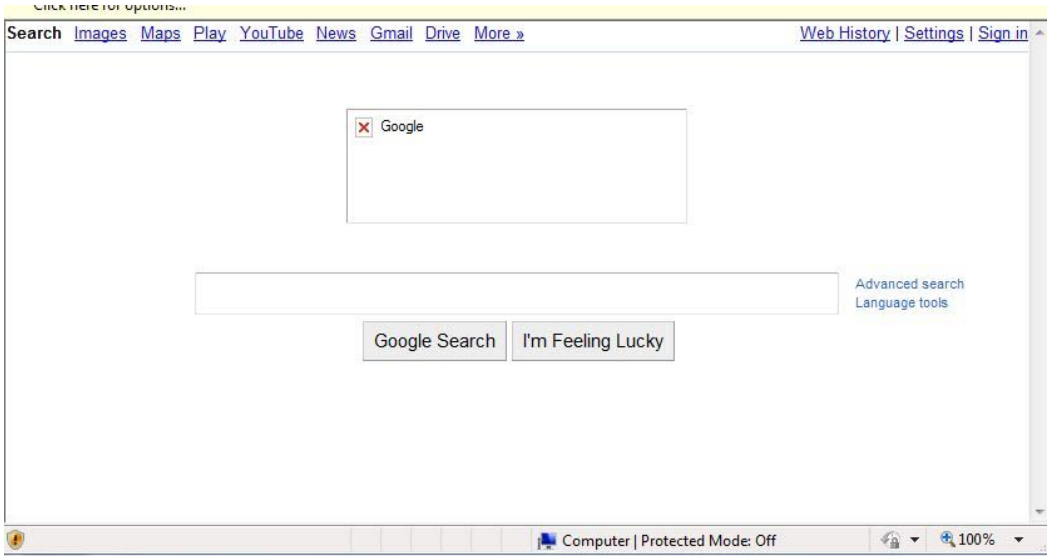
STEP 1: Create the URL with Http URL Connections
STEP 2: Define the Http Protocol for Client Connections.
STEP3: Get the Http Connection.
STEP4:Print the URL for the Client.

Program:

```
 import java.io.*;
import java.net.*;
public class myhttp
{
public static void main(String args[])throws IOException
{
URL url=new URL("http://www.google.com/");
URLConnection conn=url.openConnection();
conn.connect();
InputStreamReader content= new InputStreamReader(conn.getInputStream());
FileWriter f=new FileWriter ("abc.html");
for(int i=0;i!=-1;i= content.read())
{
f.write((char) i);
}
}
}
```

Output:





Result: The above program is executed and output was verified.

Expt. No. 07	Page No. 22
Expt. Name. Creation of Telnet Protocol	Date. 05/09/18

Aim: Program to perform telnet protocol using java.

Algorithm:

SERVER:

STEP1. Declare the header file and create class name.

STEP2. Create an object in the server which check where its connection is established or not.

STEP3. Now send the message as input to client.

STEP4. Print the message

CLIENT:

STEP1. Declare the header file.

STEP2. Create a class name for class.

STEP3. By using buffer condition are read and print the input.

STEP4. Display the connection message which is given in server.

Program:

Server:

```
import java.lang.*;
import java.io.*;
import java.net.*;

class TelnetServer {
  public static void main(String args[]) {
    String data = "Hello Client!! ";
    try {

      ServerSocket srvr = new ServerSocket(8088);
      Socket skt = srvr.accept();
      System.out.println("Client Connected!");
      PrintWriter out = new PrintWriter(skt.getOutputStream(), true);
```

```
                DataInputStream din=new DataInputStream(skt.getInputStream());
                if(din.readUTF().equals("1")){
                System.out.println("String: '" + data);
                out.print(data);
                }
                out.close()
                skt.close();
                srvr.close();
                din.close(); /
              }
              catch(Exception e) {
                System.out.print(e);
              }
            }
          }
```

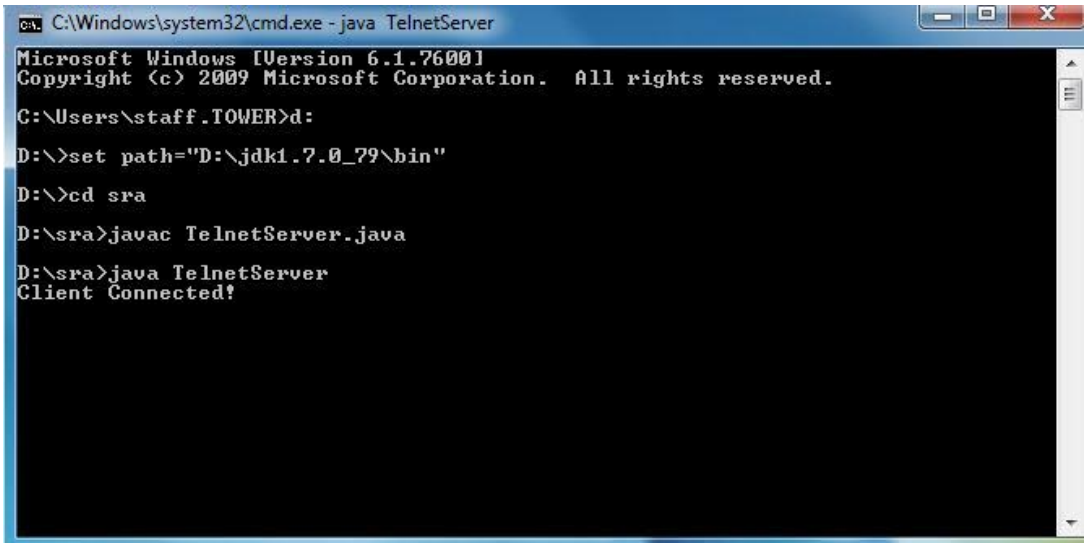
Client:

```
          import java.net.*;
          import java.io.*;
          class TelnetClient
          {
             public static void main(String args[]) throws Exception
             {

                Socket soc=new Socket("localhost",8088);
                String Command;
                DataInputStream din=new DataInputStream(soc.getInputStream());
                DataOutputStream dout=new DataOutputStream(soc.getOutputStream());
                BufferedReader br=new BufferedReader(new InputStreamReader(System.in));
                System.out.println("Welcome to Telnet Client");
                System.out.println("< Telnet Prompt >");
                Command=br.readLine();
                dout.writeUTF(Command);
                System.out.println(din.readLine());
                soc.close();
```

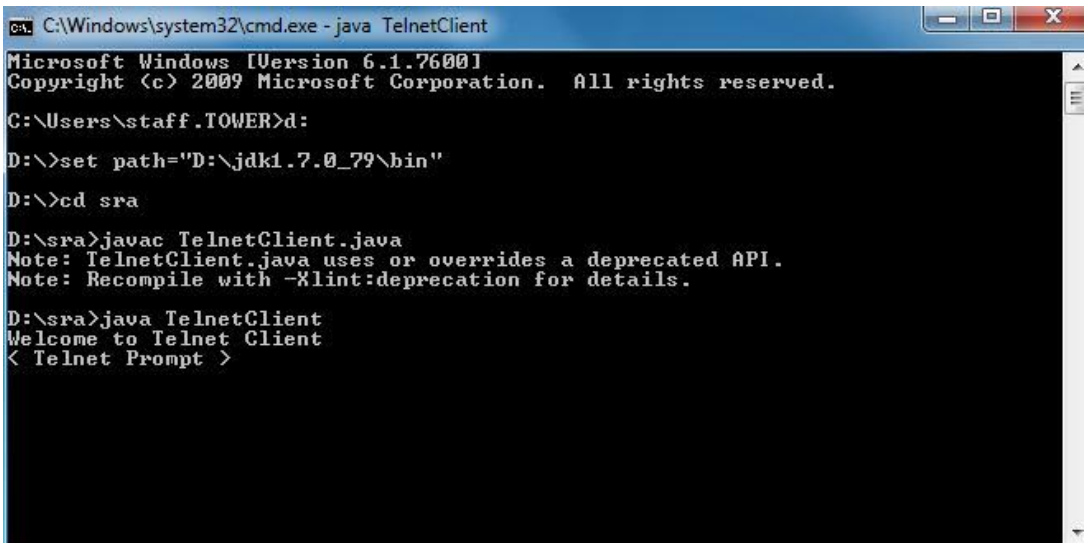
```
            din.close();
            dout.close();
            br.close();
        }
    }
```

Output:

Server:



Client:



Result: The above program is executed and output was verified.

Expt. No. 08	Page No. 25
Expt. Name. Implement FTP using TCP	Date. 05/09/18

Aim: Program to implement FTP using TCP.

Algorithm:

CLIENT

STEP 1: Create instance for the Socket class and establish connectivity with the server

STEP 2: Use the port number 4000
STEP 3: Receive the file from the server
STEP 4: Reset the connection with the server

SERVER

STEP 1: Create instances for the serversocket class and accept the server port
STEP 2: Read the filename to be opened
STEP 3: Send the file to the client

Program:

server:

```
import java.net.*;
import java.io.*;
public class ftpserver extends  Thread
{
public static void main(String args[])
{
try
{
ServerSocket ss=new ServerSocket(4000);
Socket s=ss.accept();
BufferedReader br=new BufferedReader(new
InputStreamReader(s.getInputStream()));
BufferedReader in=new BufferedReader(new InputStreamReader(System.in));
PrintWriter out=new PrintWriter(s.getOutputStream(),true);
String fn,contents=" ",temp;
System.out.println("enter the file name to open: ");
```

```
        fn=in.readLine();
        File f=new File(fn);
        if(f.isFile()&&f.canRead())
        {
        BufferedReader fil=new BufferedReader(new FileReader(fn));
        while((temp=fil.readLine())!=null)
        contents=contents+temp+"\n";
        }
        else
        contents="error in input file";
        System.out.println(" the contents of the file is:\n"+contents);
        System.out.println("sending the file to the client....");
        out.println(contents);
        }
        catch(Exception e)
        {
        System.out.println(e);
        }
        }
        }
```

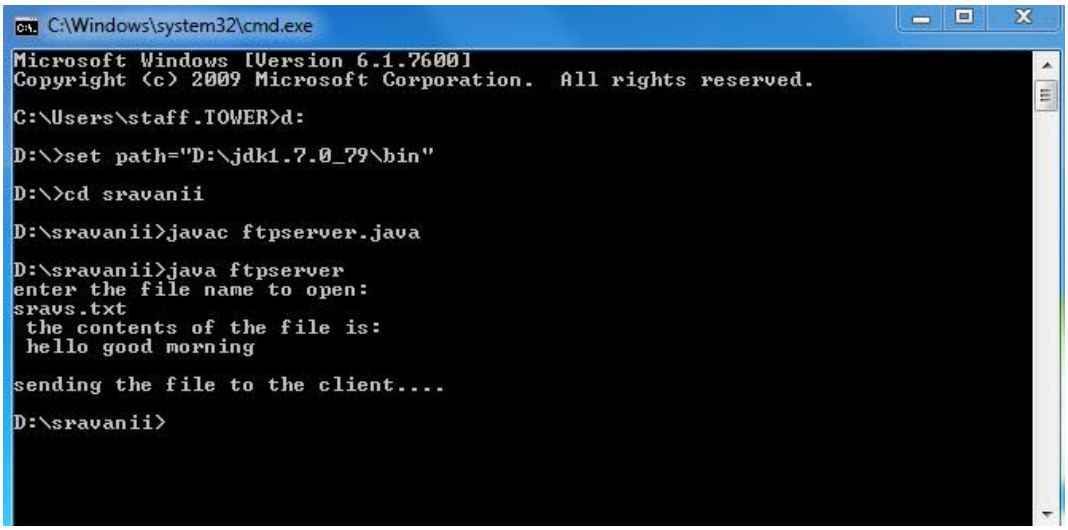
Client:

```
        import java.net.*;
        import java.io.*;
        public class ftpclient   {
        public static void main(String args[])
        {
        try
        {
        Socket s=new Socket(InetAddress.getLocalHost(),4000);
        BufferedReader br=new BufferedReader(new
        InputStreamReader(s.getInputStream()));
        String str;
        while((str=br.readLine())!=null)
        System.out.println(str);
```

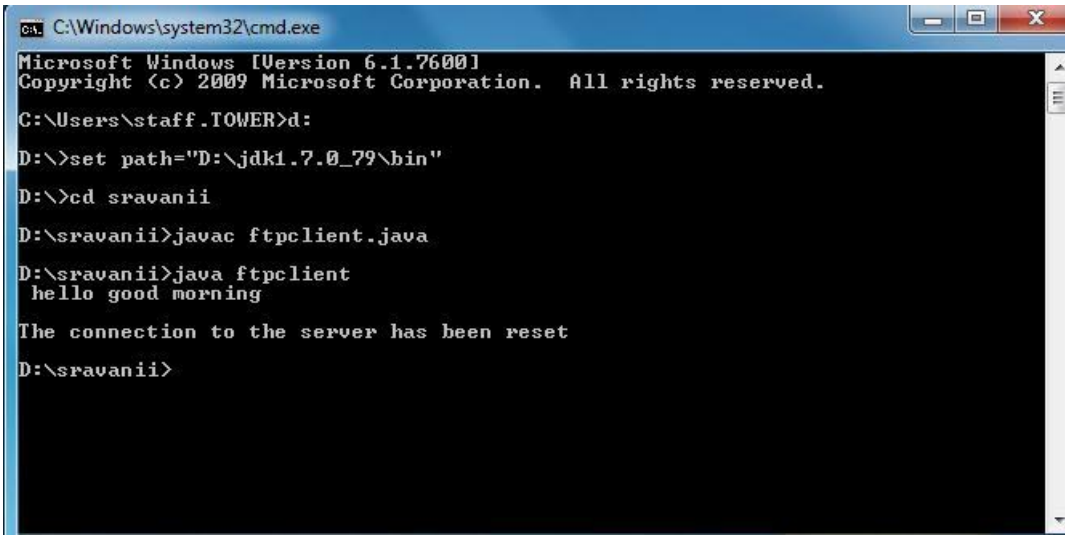
```
}
catch(Exception e)
{
System.out.println("The connection to the server has been reset");
}
}
}
```

Output:

Server:



Client:



Result: The above program is executed and output was verified.

Expt. No. 09	Page No. 28
Expt. Name. Implement FTP using TCP	Date. 05/09/18

Aim: Program to implement FTP using UDP.

Algorithm:

CLIENT

STEP 1: Create instance for the Socket class and establish connectivity with the server

STEP 2: Use the port number 8000.
STEP 3: Receive the file from the server
STEP 4: Reset the connection with the server

SERVER

STEP 1: Create instances for the serversocket class and accept the server port
STEP 2: Read the filename to be opened
STEP 3: Send the file to the client

Program:

server:

```
import java.net.*;
import java.io.*;
public class Server
{
        public static void main(String[] args) throws Exception
        {
        InetAddress sa=InetAddress.getByName(null);
BufferedReader in=new BufferedReader(new InputStreamReader(System.in));
String msg="",fn,tmp;
System.out.println("Enter the File name:");
fn=in.readLine();
File f=new File(fn);
if(f.isFile() && f.canRead())
        {
        BufferedReader fil=new BufferedReader(new FileReader(fn));
        while((tmp=fil.readLine())!=null)
```

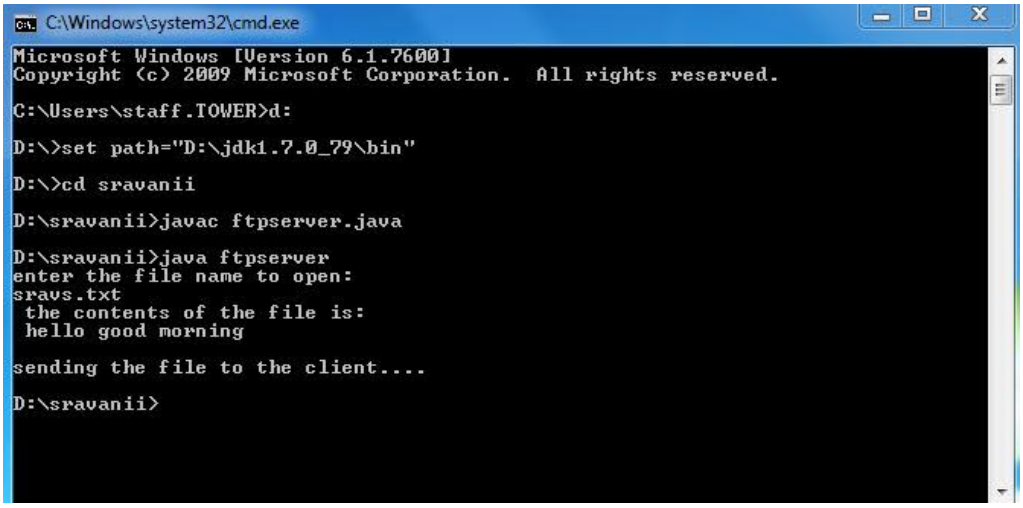
```
            msg=msg+tmp+"\n";
            }
        else
            msg="ERROR IN IMPUT FILE";
            System.out.println(msg);
            byte data[]=new byte[msg.length()];
            msg.getBytes(0,msg.length(),data,0);
            DatagramSocket ds = new DatagramSocket(8000);
            DatagramPacket dp=new DatagramPacket(data,data.length,sa,8001);
            ds.send(dp);
            }
        }
```

Client:

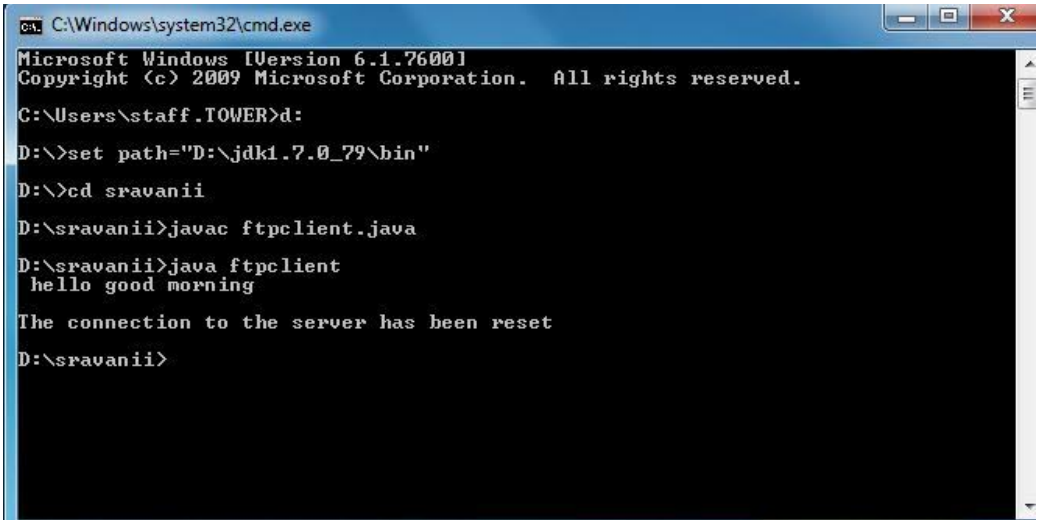
```
        import java.net.*;
        import java.io.*;
        public class udpclient
        {
            public static void main(String[] args) throws Exception
            {
            InetAddress sa=InetAddress.getLocalHost();
            byte data[]=new byte[1024];
            DatagramSocket ds = new DatagramSocket(8001);
            DatagramPacket dp=new DatagramPacket(data,data.length);
            ds.receive(dp);
            String msg=new String(dp.getData(),0,0,dp.getLength());
            System.out.println("Received data : " + msg);
        }
        }
```

Output:

Server:



Client:



Result: The above program is executed and output was verfied.

Expt. No. 10	Page No. 31
Expt. Name. Router Configuration	Date. 05/09/18

Aim: Study of Router Configuration Commands.

Procedure:

1) To enter privileged EXEC, enter the enable EXEC command.

Router> **enable**

Password:

Router#

2) To exit privileged EXEC mode, enter the disable EXEC command.

Router# **disable**

Router>

3) To enter Global Configuration mode, enter the Configure Terminal command.

Router# **configure terminal**

Router(config)#

4) To end the current configuration session and return to privileged EXEC mode, use the end global configuration command.

Router(config)# **end**

Router>

5) The show interfaces command will show the information about all the interfaces

Router# **show interfaces**

6) A show command is used in privileged EXEC mode to verify the configuration.

Router# **show interface serial 0/1/0**

7) the exit (global) command is used to move from global configuration mode to privileged EXEC mode, the disable command is used to move from privileged EXEC mode to user EXEC mode, and the logout command is used to log off

Router(config)# **exit**

Router# **disable**

Router> **logout**

8) The command hostname is used to specify the name for the router.

```
Router(config)# hostname Router
Router(config)#
```

9) The command enable secret password is used to specify an encrypted password to prevent unauthorized access to the router.

```
Router(config)# enable secret cr1ny5ho
Router(config)#
```

10) The Interface command is used to enter into the configuration mode for a Fast Ethernet WAN interface on the router.

```
Router(config)# interface fastethernet 0
Router(config-int)#
```

11) The command ip address is used to set the IP address and subnet mask for the specified Fast Ethernet interface.

```
Router(config-int)# ip address 192.1.12.2 255.255.255.0
Router(config-int)#
```

12) The no shutdown command enables the Ethernet interface, changing its state from administratively down to administratively up.

```
Router(config-int)# no shutdown
Router(config-int)#
```

Result: Study of Router Configuration Commands was completed.